SmartEx: A Framework for Generating User-Centric Explanations in Smart Environments

Mersedeh Sadeghi, Lars Herbold, Max Unterbusch, Andreas Vogelsang

University of Cologne, Cologne, Germany

sadeghi@cs.uni-koeln.de, lars.herbold@outlook.de, munterbu@smail.uni-koeln.de, vogelsang@cs.uni-koeln.de

Abstract—**Explainability is crucial for complex systems like pervasive smart environments, as they collect and analyze data from various sensors, follow multiple rules, and control different devices resulting in behavior that is not trivial and, thus, should be explained to the users. The current approaches, however, offer flat, static, and algorithm-focused explanations. User-centric explanations, on the other hand, consider the recipient and context, providing personalized and context-aware explanations. To address this gap, we propose an approach to incorporate user-centric explanations into smart environments. We introduce a conceptual model and a reference architecture for characterizing and generating such explanations. Our work is the first technical solution for generating context-aware and granular explanations in smart environments. Our architecture implementation demonstrates the feasibility of our approach through various scenarios.**

I. INTRODUCTION

Explainability has become a vital research topic due to the recent progress in machine learning, autonomous systems, complex decision-making, and smart cyber-physical systems [1], [2], [3], [4], [5], [6], [7]. Users can lose trust in a system if they feel they have no control and if the system's behavior does not meet their expectations, leading to misuse or rejection [8], [9]. Many studies indicate that providing explanations is a viable solution because it enables users to develop a more accurate mental model of the system, allowing them to anticipate and interpret its behavior more effectively [5], [10], [11], [12].

The advancements of smart devices, cloud computing, mobile networks, and pervasive computing have resulted in highly complex ecosystems [13]. These ecosystems gather extensive data on the environment and users' actions, goals, and preferences, facilitating context-aware decision-making and autonomous activities. The ultimate goal of this pervasive intelligence is to enhance user satisfaction and comfort. Nevertheless, the autonomous and automated nature of these systems can often lead to confusion and frustration among users [5]. Subsequently, there is a need to provide explanations for users at runtime to clarify the cause, purpose, and benefit of the system's actions.

We've identified issues with current explanation approaches in the smart environments domain. Firstly, many of them focus solely on algorithmic explanations [14], [15], [16], which describe system behavior in terms of algorithms and processes [15], [17], [18], [19]. These explanations often unravel the planning or a causal chain of events, that are challenging for end-users to comprehend [20], [21], [22]. Furthermore, algorithmic explanations neglect the user's context. They provide uniform explanations, overlooking other explanation desiderata besides describing the inner workings of a system, such as persuasion, learning, or assignment of blame [23], [24]. The algorithmic explanation constitutes an objective explanation, that only considers the problem to be explained. However, the other desiderata require a more subjective explanation, that considers the context in addition to the problem. Hence, the user-centric explanation should encompass contextual elements, reasoning, and information beyond cause-effect relations. [16], [20]. Finally, existing approaches for explanation generation in smart environments often suffer from a lack of personalization. Bunt et al.'s empirical study [25] shows differing perceptions of explanation attributes like completeness, soundness, and complexity. This suggests that a one-fits-all explanation is insufficient, and there is a need for personalized and granular explanations that consider individual differences. Despite wide agreement on the importance and essential attributes of user-centric explanations, we lack approaches for developing smart environment systems that can generate them.

To address the aforementioned shortcomings, we propose building an explanation layer on top of a smart environment system that constructs user-centric explanations by leveraging the capabilities inherent in such systems (e.g., physical sensors and devices, data collection and processing components, and knowledge extraction mechanisms). Our approach offers the following contributions:

- We define a model for user-centric explanations in smart environments.
- We present a reference architecture for a user-centric explanation generation engine, identifying event causes within a rule-based system and enhancing explanations with pertinent context to deliver personalized, context-aware explanations in natural language.
- To assess the feasibility of our solution, we have implemented the proposed reference architecture.

II. RELATED WORK

Ambient smart environments proactively support people in their daily lives [26]. According to Huang et al. [27], a mismatch between a user's mental model and a system's logical interpretation can result in program failure and reduced perceived ease of use. Using explanations effectively addresses the issue by enhancing system transparency, improving user mental models, and ultimately boosting trust and usability [5], [10], [12]. Many studies emphasize the significance of experiential, explorative, contextual, and timely explanations to improve user interpretability; adding a human touch can increase social presence [16], [28], [29], [30]. Miller [20], Hoffman et al. [31], [32], Klein [33], and Mittelstadt et al. [34] in their studies have overviewed decades of works on social sciences for

explanation attribution. According to their finding, effective explanations must be user-centric, i.e., cater to users' epistemic states, offering both "everyday" and scientific explanations. Paes [35] discusses that explanations need to convey both a pragmatic and naturalistic account of understanding, and Gregor and Benbasat [36] reason that explanations for end-users must be context-based.

Moving to Software Engineering for Explainable Smart Environments, we can refer to some related works. FORTNIoT [37] is a smart home framework that offers a self-sustaining prediction model and simulations for forecasting rule occurrences in smart homes. Additionally, it provides explanations for future entity changes, helping users identify triggering rules and their reasons. Unlike our research, FORTNIoT mainly focuses on forecasting events and provides limited explanations that show cause-effect relations among rules, lacking contextual and personalization aspects. Houz'e et al. [38] developed an explainable smart home framework, comprising explanatory components for specific types of sensors and smart devices. Like our approach, they see smart home dynamics as predicates and trace events to pinpoint which predicates support the questioned proposition, constituting the explanation. However, their architecture solely deals with simple rule-based systems where individual objects have their rules (e.g., a thermostat turning off if the temperature exceeds a threshold) and don't delve into complex rules involving group devices, predicates, and events. Moreover, objects must adhere to standard interfaces to integrate into their framework, which limits their effectiveness due to the wide diversity of smart devices. Authors in a set of works [39], [40] focused on developing a loosely coupled mechanism for generating explanations. They introduced a description language for smart devices and a mediator acting as middleware to handle the diversity of sensors and smart objects. Their unique contribution lies in incorporating information about device functions into the description, enabling smart devices to provide self-explanations. In contrast to our approach, their work primarily aims at offering tutorials to help users better operate devices. In our work, our primary focus is on enabling intelligent systems to self-explain automated actions and decisions to enhance human-computer interactions and engagement for end-users. In a related vein, there is a substantial body of research pursuing a similar objective in a broader application domain, like cyber-physical systems. Blumreiter et al. [41] introduced MAB-EX (Monitor, Analyze, Build, Explain), a reference architecture for generating explanations based on the MAPE-K framework for self-adaptive systems [42]. In their system, an explanation takes the form of a behavioral model of the system, capturing causal relationships between events and system responses. The leaves in their cause tree are linked to static explanations that can ultimately be presented to the user. While the authors discussed the potential for automated explanation creation, their initial demonstration relied on manual explanation generation.

To conclude this section, we can observe that the focus in explainability domains has mainly revolved around algorithmic explanations. These explanations mainly hinge on causal relations, presented as diagrams, charts, or stats, which can be challenging for end-users. Hence, there is a growing imperative to shift towards user-centric [43], [44]. In our work, we start with causal explanation generation but then extend it with contextual elements by utilizing the inherent capabilities of a smart environment. Indeed, smart environments, comprising interconnected sensors, devices, and mechanisms, offer valuable resources for user-centric explanations. Despite the potential of context-awareness in enhancing various aspects of ambient environment systems, such as personalization, location-based services, and activity recognition [45], [46], [47], [48], explainability in smart environments is often overlooked, as highlighted in related works [38], [39]. Our research aims to bridge this gap by offering a practical framework for creating user-centric explainable systems.

III. User-Centric Explanations Conceptualization

A. Motivating Scenario

We devised a scenario inspired by a collection of situations that need to be explained in an interactive smart environment proposed in [49]. The scenario involves three users—Alice, Bob, and Dana—interacting in an office building equipped with smart devices and Smart Office Manager (SOM) software, controlling various intelligent functionalities. Bob has set up a rule to mute the TV automatically whenever a meeting occurs in rooms located near the kitchen. Sometime later, Alice tries to play music during lunch, but the TV mutes itself, leading to her frustration. Despite her attempts to unmute it, the TV keeps muting itself. Alice needs an explanation for this behavior. Bob sees Alice struggling with the TV but does not realize he set a rule to mute it during nearby meetings, especially since he is unaware of the meeting in Room 1. Now, he needs an explanation too. On another occasion, Dana, a visitor, is waiting in the kitchen for her host Chuck, and experiences the TV suddenly going silent. Despite not actively watching the TV, the abrupt change in the TV audio left Dana confused, particularly because she is alone in the kitchen and has not interacted with any system. As a result, she feels the need for an explanation.

In this scenario, three individuals find themselves in a perplexing situation. However, as we will elaborate in the upcoming section, each individual requires a personalized explanation distinct from the others. Moreover, the explanation must be easily accessible to them through the same smart infrastructure they use for managing the intelligent environment. This pervasive provision of explanation is crucial for facilitating user interaction.

B. Explanation Conception

Before presenting our explanation formalization in Definition 1, we provide a brief overview of the notations used throughout this work. The term *Smart Environment System S* defines modern smart homes, offices, and buildings, as having interconnected sensors and intelligent devices communicating with each other and users. They typically possess autonomous perception, reasoning, and action capabilities in their environment. Additionally, the term **Explanandum** ϕ refers to the state or behavior of a system S that an **Explainee** requests an explanation for. An ***Explainee***, in our study, is a human who interacts with the system S. Building on the causal explanation framework introduced in [50], we broaden its application to

TABLE I
EXAMPLE OF ALGORITHMIC AND SEMANTIC CONSTRUCTS IN OUR MODEL

Explanation Construct	Example
Algorithmic Explanation Constructs (for causal explanation)	(χ_r: $Rule_k$ is fired) $\rightarrow$ (χ_s: TV is Mute) (χ_p : Meeting M1 is going on in Room 1) $\wedge$ (χ_q: TV is On) $\rightarrow$ (χ_r: $Rule_k$ is fired)
Contextual Explanation Constructs (for context-based explanation)	(χ_i : Bob has set rule $Rule_k$)$\wedge$ (χ_j : $Rule_k$ states that if TV is on while a meeting is taking place in rooms 1, 2, or 3, then mute TV)

encompass user-specific and contextual factors overlooked in the generic approach. The original formulation proposes a causal explanation of a phenomenon *P* through a conjunction of primitive *Events*. This method offers the advantage of a simple and generic structure for composing explanations by identifying a sequence or combination of events that led to the occurrence of *P*. However, the main drawback is that it lacks consideration for the user and contexts resulting in a mechanistic explanation that may not address the user's needs. Therefore, our objective is twofold. Firstly, we aim to apply the concept of the generic causal explanation formalization presented in the existing literature to the context of smart environments. Secondly, we intend to extend this concept by integrating contextual aspects. To facilitate this, we introduce the notion of **Explanation Constructs** denoted as *X*.

In our model, the *Explanation Constructs* encompass a set of specifications, facts, propositions, and events related to both the internal elements *S* and the external world which includes the user's states and external systems, such as a list of smart objects managed by *S*, a general description of such devices, their current and past states, sequences of actions and events, a set of rules and specifications, a set of contextual variables (e.g., user's roles) and their current values. *Explanation Constructs* hence are among the data that are commonly accessible and processable in modern smart environments through APIs, documentation, and logs of events provided either by *S* itself or external systems. Finally, to proceed with the *Explanation* definition, we can broadly classify *Explanation Constructs* into two categories as shown in Table I.

The first category is **Algorithmic Explanation Constructs (AEC)**, which consists of facts, propositions, specifications, or events that describe the logic or mechanism behind an action/output of a machine/program. The second category is **Contextual Explanation Constructs (CEC)**, which includes facts, propositions, or specifications that provide information related to an action/output of a machine /program or an *AEC*. By incorporating both AECs and CECs in an explanation, we can create personalized and context-aware explanations. It goes beyond the traditional causal explanations that only describe the underlying mechanism of an *explanandum*. We define the notion of *Explanation* in the domain of an smart environments in Definition 1.

Definition 1 (**Explanation Ψ**). For a given ϕ, there is a Ψ that is a conjunction of some explanation constructs:

$$\Psi = (\chi_1 \in X) \wedge \ldots \wedge (\chi_k \in X) \ where \ X = \{AEC \cup CEC\}$$

Hence, given the nature of *X* and Definition 1, an *explanation* is interpreted as a piece of causal and contextual information that describes the algorithmic and contextual reasons behind

an *explanandum*. Therefore, it is aligned with the user-centric explanation characteristics remarked earlier. More concretely, an explanation is partly constructed by a causal explanation for the system's behavior captured through *AEC*. As shown in Table I, for the motivating scenario given in Section III-A, where the *explanandum* is : [ϕ: *why is the TV muted?*], the higher level *AEC* can be framed as: [*the TV is muted as a direct result of firing* $Rule_k$]. If one desires to explore the causation path in more depth[1], then *AEC* can also include: [*because a meeting in a nearby room is going on* (χ_p in Table I) *and TV is on* (χ_q in Table I)] since such proposition implies that $Rule_k$ (χ_r in Table I) fires: [$\chi_p \wedge \chi_q \rightarrow \chi_r$]. Furthermore, the complementary part of the explanation in our model is the *CEC* that provides further meta-information. In our example, this includes the rule description itself and a rationale as to why such rules exist (see χ_j and χ_i in Table I).

Our explanation model includes two further essential elements: *Views* and an *Inference Function*. Together, these elements enhance the personalization and context-awareness of an explanation going beyond a one-fits-all approach. By different *Views*, we can tailor an explanation to specific user needs and contexts. The *Inference Function* determines how the system selects a particular *View* to be shown to a particular user based on relevant contextual information. *Transformation Function* ensures that the explanation is understandably presented to the end-user by transforming the formal model of the explanation to a natural language representation.

Definition 2 (Views of Ψ). Let Ψ be the *explanation* for ϕ. There are *n* different $Views = \{V_1, V_2, \ldots, V_n\}$ of Ψ, which are combinations of $AEC, CEC \subseteq \Psi$ representing different partial *explanations* for ϕ. Note that the most detailed and comprehensive view (V_{full}) includes all $AEC, CEC \subseteq \Psi$, which essentially means $V_{full} = \Psi$.

Definition 3 (Inference Function). There exist some mechanisms (e.g., mapping, inference, or rule engines) that operate by considering a specific *explanandum* and all pertinent contexts to deduce the *most suitable* $V_i \in Views$ for presentation to a particular *explainee* within a specific situation.

Definition 4 (Transformation Function). A function that translates a view $V_i \in Views$ (and its contained AEC and CEC) into a natural language representation.

In our toy example, we say that $\Psi = \{\chi_p \wedge \chi_q \wedge \chi_r \wedge \chi_s \wedge \chi_i \wedge \chi_j\}$ (see Table I). Given the epistemic state of Bob, who has set the rule R_k, a relevant explanation is *Inference* (explainee, Contexts, Ψ) = V_1, where $V_1 = \{\chi_p \wedge \chi_q\}$ (see Table I). That is because Bob is aware of the existence

---

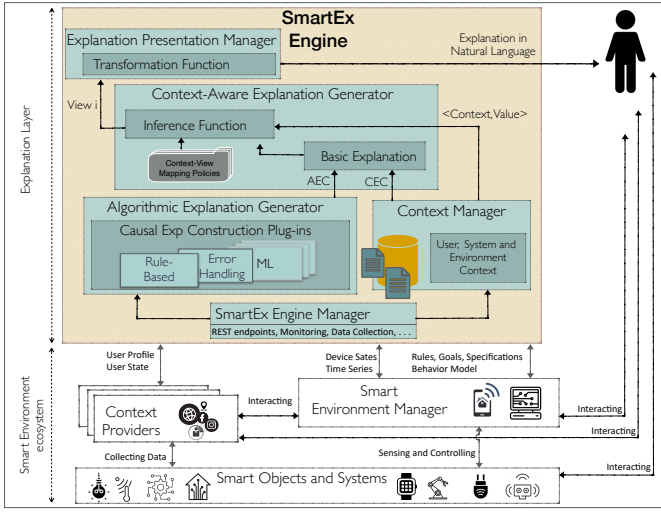[1]The *explanandum* may also be: [ϕ: *why has this rule been fired?*]

Fig. 1. Reference architecture for the Smart Explanation Engine (*SmartEx*)

of $Rule_k$. Thus by χ_p and χ_q, he can infer that the rule has been triggered. This representation is ultimately presented to Bob as Transform $(V_1) = \{Because\ Meeting\ M1\ in\ Room\ R1$ *is going on and TV is turned on}*. However, to personalize the same Ψ for Alice, as a colleague of Bob, there is a need to additionally explain the rule. Therefore, given *Inference (*explainee, Contexts, Ψ), a relevant explanation could be $V_2 = \{\chi_p \wedge \chi_q \wedge \chi_j\}$. Similarly, for Dana, who is a guest visitor and Bob prefers to avoid revealing the underlying details for her, yet another representation $V_3 = \{\chi_s \wedge \chi_i \wedge \chi_r\}$ must be synthesized to customize the original explanation. It is then offered to Dana as Transform(V_3) = {*TV is muted because Bob has set a rule which has been just fired.*}.

IV. A USER-CENTRIC EXPLANATION ENGINE

Figure 1 shows the reference architecture for the Smart Explanation Engine (*SmartEx*), a RESTful service to extend typical smart environment systems to an explainable ecosystem. It decouples the explanation generation process from the core intelligent system, which includes components like the smart environment management system and home automation hub, as well as intelligent device services and applications.

To create user-centric explanations, the *Context-Aware Explanation Generator* component needs to have the current usage contexts supplied by the *Context Manager* and the *algorithmic explanation* which is the output of the *Algorithmic Explanation Generator* component. The *Context Manager* collects and preprocesses contextual elements. Some, like the *Time* context, can be obtained directly on request, while others, such as relatively stable user details (e.g., name, age), are stored locally for future access. Additionally, it can utilize external context providers (e.g., smart environment APIs, social networks) for more complex and dynamic contexts. The supplied contextual elements will then be used by the *Inference Function*. It is responsible for mapping the current values of the relevant contexts to a particular *View* (see Definitions 2 and 3) among the set of *Views* known by the system. Such a view will ultimately be translated into natural language, which is understandable for a human interacting with the system.

The implementation of the *Algorithmic Explanation Generator* and the set of *Views*, depends on intelligent methods deployed in the systems, such as various decision models (e.g., rule-based, hidden Markov model, decision tree), recommendation systems (collaborative and content-based filtering), path planning, or ML systems. In this work, we are targeting the Rule-based systems (see Section IV-A), which are among the most popular approaches for building smart environments and context-aware systems [51], [52]. However, new plug-ins of explanation generations tailored for other intelligent methods can be integrated into the system in future works.

A. Explanations for Rule-based Systems

Each smart object in a typical rule-based system in the smart environments has *Actions*, which are operations performed by the object, and *Properties* that describe the internal states of the object. A business *Rule* in its most generic form can be defined as an expression: **If** *Precondition*, **then** *Action*. In smart environment systems, rules serve as proxies to trigger specific *Actions* for smart objects when certain preconditions (e.g., specific property values or contextual factors) are fulfilled.

We can conceptualize a rule-based smart environment system S as a knowledge graph, where each node represents a conceptual or physical element, such as a smart object, its actions and properties, rules, and environmental contexts within the system S. Starting from an *Action* node in this graph, multiple paths unfold, representing plausible causes for the execution of that specific action. For instance, in our motivating scenario, the "mute TV" action may be issued due to a fired rule (from a set of rules $R_1, \ldots, R_n$), a direct command via an API, or due to remote control signals. Each of these possibilities forms a path in the graph, with the root node being "mute TV". Each rule ($R_1, \ldots, R_n$) involves distinct preconditions, combining various smart object properties, user actions, and environmental context. In reality, among all these paths, only one path exists that represents the real cause of an action. [2] This unique path represents the cause-and-effect chain. In our example, this path comprises preconditions (TV is on and there is a meeting nearby), leading to the fired rule (say, R_1), resulting in the "mute TV" action.

Our framework has information about the rules of a system including their preconditions and actions. By comparing the rule descriptions with observed events extracted from system logs, we can construct a cause-and-effect chain for an action. To achieve this, our system utilizes a mechanism (see Section IV-A1) to determine the causal path behind a system behavior. After constructing the causal explanation component, *SmartEx* further refines the explanation considering relevant contextual factors (see Section IV-A2).

1) Algorithmic Explanation: Algorithm 1 shows how we identify the cause-and-effect path discussed above. It takes two inputs. The first is the *Action* to be explained (i.e., the *explanandum*). The second is a set of *Explanation Constructs* X, which include a list of smart objects managed by S, a general description of such devices, their current and past states, sequences of actions, and events, i.e., the log of S from m minutes ago until the time the explanation was requested.

[2] We assume no duplicate rules and no rule conflicts.

The output of the algorithm is either the unique path that results in the *explanandum* or *Null* if no such path exists.

In essence, the path corresponds to the rule that has been triggered, resulting in the execution of the *explanandum* (*FiredRule* in Alg. 1), along with all preconditions and actions associated with that rule. To ascertain the output, the algorithm initiates by extracting and removing R from the set X. R represents the collection of rules established by users. The remaining constructs in X are then sorted in order from T_{now} (when the request for generating an explanation is made) back until T_m. The variable m is configurable and determines the extent to which the algorithm examines past events.
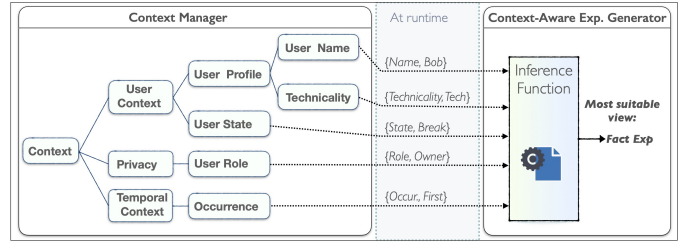

Fig. 2. Context model (left) and its usage at runtime. The Context Manager provides the Inference Function with the current values of relevant contexts. In this example, the Inference Function infers that the most appropriate view to present to the user is the Fact Explanation View.

Algorithm 1 Find the cause path

1: **procedure** FIND THE CAUSE PATH P
2: **input:** X: Set of all explanation constructs observed by the system (i.e., rules, actions, preconditions, logs of events, etc.), $explanandum$: the explanandum
3: **output** path: P
4:     $P \leftarrow \emptyset$
5:     $FiredRule \leftarrow \emptyset$
6:     $R \leftarrow$ Set of rules extracted from X
7:     $X \leftarrow X - R$
8:     **sort** $x \in X$ in reverse chronological order from t_{now} to t_m
9:     $CandidateRules \leftarrow r$ where $\{r \in R \wedge explanandum \in r.actions\}$
10:    $FiredRule \leftarrow$ **find** $r \in CandidateRules$ where $\{$
11: $(\forall a \in r.actions : (a \in X \ \wedge \text{time}(a) = \text{time}(explanandum))) \ \wedge$
12: $(\forall p \in r.preconditions : (p \in X \ \wedge \text{time}(p) < \text{time}(explanandum))) \ \wedge$
13: $(\text{eval}(r.preconditions, X, \text{time}(explanandum)) = \textbf{true})\}$
14:    **if** $FiredRule \neq \emptyset$ **then**
15:       $P \leftarrow FiredRule \wedge FiredRule.preconds \wedge FiredRule.actions$
16:    **return** P

Subsequently, the algorithm searches through the rules to identify a subset of R, which is referred to as *CandidateRules*. This subset includes rules that contain the *explanandum* as an action(*r.actions* in Alg. 1, line 9). Next, the algorithm assesses each element of the *CandidateRules* by traveling back through the log of events in X. The aim is to locate the specific rule r that satisfies two conditions(lines 10-13): i) all of the other actions linked to r must also have been triggered simultaneously with the *explanandum* and ii), all of the preconditions associated with r must have been satisfied before the *explanandum*. To evaluate these conditions, it is sufficient to check the existence of the required preconditions and actions in X and their timing order through a function that returns the timestamp of X.

Please note that a rule may have multiple actions, which are all executed when the rule's required preconditions are met. The actions of rule r are grouped using the *AND* operator, meaning that **all** of r's actions must have been triggered. Unlike actions, preconditions of a rule can be combined using both the *AND* and *OR* operators. To determine the overall truth value of the rule's preconditions, the *eval* function evaluates the logical expression formed by the logical operators of the preconditions. It takes preconditions of r, along with X and all timestamps, and outputs a Boolean value of *True* if all necessary preconditions are met or *False* otherwise. In short, it evaluates the innermost groups of preconditions first, assigning a truth value to them based on their operator. It then computes the truth value of the outer groups until the entire expression is resolved to a single truth value.

2) User-centric Explanation: Given Definition 1, a user-centric explanation necessitates the incorporation of *AECs* coming from *Algorithmic Explanation Generator*, and *CECs*. It generates a comprehensive explanation, which can be decomposed into multiple views to provide granular and personalized explanations (see Definition 2 Furthermore, the inclusion of a context-aware mechanism becomes imperative to determine the appropriate view for a given user in a specific situation (see Definition 3 and Algorithm 2).

In particular, we are introducing four different views for a user-centric explanation Ψ in a rule-based system. The first view is Full Explanation View, corresponds to the complete Ψ that is composed of both *AEC* (i.e., the path P) and *CEC* (i.e., the rule definitions and the creators of rules). The subsequent views then offer partial representations of Ψ by adjusting its granularity and length to cater to different user needs. In particular, the Rule Explanation View only denotes the rule $r_i \in R$, whose action needs explanation (i.e., the root of the path P). The Fact Explanation View displays the set of events that fulfill the preconditions of $r_i \in R$, resulting in the rule's activation. It provides a representation of the instances of the preconditions in the real world, thereby explaining why the rule has been activated. On the other hand, the Simplified Explanation View offers a high-level explanation for an *explanandum* by simply stating that the event occurred due to a rule (without revealing the rule itself) set by a specific user u_i.

The final procedure to provide a context-aware and personalized explanation is to map a relevant set of contextual elements to the most suitable view. In our work, contexts are modeled as tuples of *(attributeName, attributeValue)* pairs. We incorporate only relevant contexts for an explanation tailoring, meaning alterations in their value may impact the explanation's length and granularity. Thus, our context model is composed of a non-exhaustive set of contexts that we have found to be pertinent for preferring one explanation over another. More specifically, our system currently relies on some specification of three main general contexts (User, Privacy, and Temporal Context) as shown in Figure 2.

In our context model, User Profile captures static user attributes such as User Name and Technicality Level. The dynamic attributes of the user are modeled using the User State, which provides valuable insight into the user's current situation. For instance, if the User State is Working, it indicates that the user is busy and thus may require a shorter explanation. The frequency of exposure to a certain confusing situation, referred to as the Occurrence of the *explanandum* in our model, is another significant factor affecting the level of detail required in an explanation. For example, if a user encounters a new *explanandum* for the first time, they will

TABLE II
CONTEXT-VIEW MAPPINGS.

Expr	Context / View	P1: User State			P2: Occurrence		
		Mt	Bk	Wk	Ft	St	M
1	Full Exp. View	x	✓	x	✓	x	x
2	Fact Exp. View	x	✓	✓	✓	✓	x
3	Rule Exp. View	✓	✓	✓	x	✓	x
4	Simplified Exp. View	✓	✓	✓	x	x	✓
		P3: Technicality			P4: Role		
		Tch	Med	Ntch	Ow	Cw	Gst
1	Full Exp. View	✓	✓	x	x	✓	x
2	Fact Exp. View	✓	x	x	✓	✓	x
3	Rule Exp. View	✓	✓	x	✓	✓	x
4	Simplified Exp. View	✓	x	✓	✓	✓	✓

Expr: Expressiveness, **Mt:** Meeting, **Bk:** Break, **Wk:** Working, **Ft:** First Time, **St:** Second Time, **M:** More, **Tch:** Technical, **Med:** Medium Technical, **Ntch:** Not Technical, **Ow:** Owner, **Cw:** Co-Worker **Gst:** Guest. The ✓ symbols: the view fits for the context variable. An x symbol: the view is not suitable that context.

naturally require a complete explanation compared to a situation where they have previously received an explanation for the same *explanandum*. Finally, the `User Role` context accounts for the system's privacy settings, influencing the scope and type of information that can be disclosed in an explanation.

The *Context Manager* is responsible for monitoring and retrieving current contextual element values from various sources. It supplies the *Context-Aware Explanation Generator* with two types of data. Firstly, the relevant contextual information to build Ψ (i.e., *CEC*) and secondly, the relevant contexts to infer the best-suited explanation view for the given situation. As illustrated in Figure 2, the current values of contextual elements are provided to the *Inference Function* by the *Context Manager* during the explanation generation process. The *Inference Function* uses this information to infer the appropriate explanation view to generate. This is done via Algorithm 2, utilizing contextual elements as input to determine the most suitable *View* for the situation.

Specifically, we have implemented a set of mapping policies using rules in a rule engine. Table II shows a simplified representation of them. The priorities (P1 to P4) for each context determine the evaluation order. The table shows what the suitable views are for each possible value of a given context (denoted by ✓ symbol). The table also represents the expressiveness order of views where the *Full Explanation* is the most expressive one followed by, *Fact*, *Rule*, and *Simplified*.

Algorithm 2 shows the process by which the *Inference Function* selects the appropriate view. Using the Context-View mappings and the expressiveness order of views, it calculates a *view* from the set of *Views* that is suitable for the given situation. At the outset of the process, the *Inference Function* initializes the set of available *Views* to include all possible types of views (line 4). It then sorts the policy rules in order of priority, starting with the highest (line 7). The function then begins executing each policy by evaluating the conditions specified in the rules, that is, by assessing the actual values of the relevant contexts at that time. The result of each policy execution is a set of so-called *SuitableView* (line 9). For instance, suppose a user, denoted as *U1*, is on a break, then the corresponding suitable views are *[Full, Fact, Rule, and Simplified]*. Then, the algorithm computes the intersection between the set of all views and the set of *SuitableView* for *U1*. If the intersection

is non-empty, the algorithm replaces the *Views* set with the resulting set (lines 10–11).

Algorithm 2 Inference Function

1: **procedure** FIND MOST SUITABLE VIEW FOR GIVEN CONTEXTS
2: **input:** Set of Context_View mappings, Expressiveness order of View, Current values of contextual elements
3: **output:** suitable $view \in Views$
4:     $Views \leftarrow \{FullExp, FactExp, RuleExp, SimplifiedExp\}$
5:     $Policy \leftarrow$ Set of all the $Context\_View$ mappings   ▷ see Table II
6:     $Context\_Knowledge \leftarrow \{(C_1, Value_1), \ldots, (C_i, Value_i)\}$
7: **sort** $P \in Policy$ from highest priority to lowest
8: **forEach** $P \in Policy$ **do**
9:     $Suitable\_View \leftarrow$ apply($Policy, Context\_Knowledge$)
10:     **if** $Views \cap Suitable\_View \neq \varnothing$ **then**
11:         $Views \leftarrow Views \cap Suitable\_View$
12: **sort** $view \in Views$ based on expressiveness
13: **return** most expressive $view \in Views$

Afterward, the algorithm repeats the same process for policies with the next priority, which in this case is the *Occurrence* context. Assume that *U1* is requesting an explanation for a particular *explanandum* for the *second* time. Based on the policy for the *Occurrence* context in Table II, the set of suitable views will exclude the *Full* and *Simplified Explanation*. The decision-making heuristic applied here is based on the fact that the *Full Explanation* view is deemed too detailed and is best suited for first-time explanations, while the *Simplified Explanation* view is too abstract and is better suited for situations where a user has faced the *explanandum* more than twice in the last three months. Consequently, the set of *Views* at this stage is restricted to *Rule* and *Fact Explanation*. The procedure then continues by examining the rules for the remaining contexts. As shown in Table II, our implemented *SmartEx* adheres to the context model presented in Section IV-A and includes *User Technicality* and *User Role* in addition to *User State* and *Occurrence* discussed earlier. In brief, for the *Technicality* and *User Role* contexts, the Context-View mappings assign the simplest and most abstract explanation's view to *Non-technical* and *Guest* users.

The last step is the sorting of the *Views* set based on their expressiveness by the *Inference Function*. The most expressive view is then returned as the final output. Note that the check at line 10 ensures that *Views* set will never be empty. The *Explanation Presentation* Component is responsible for translating this output to natural language through the *Transformation Function*, which could be implemented via different approaches, such as a template-based text generation or Prompt-Based Text Generation via Large Language Models.

V. IMPLEMENTATION AND FEASIBILITY CHECK

We created a prototype implementation as a proof of concept and tested it extensively in various scenarios in our own *smart environments lab*. Due to space constraints, we report results for only one scenario in this paper. The prototype is a valuable basis for conducting user studies that we plan for future research.

We have developed *SmartEx*[3] as a RESTful web service implemented in Java, using MongoDB as a database. This web service can be integrated into existing non-explainable systems to create an explainability layer on top of them. In our lab, we have an array of sensors, actuators, smart lights,

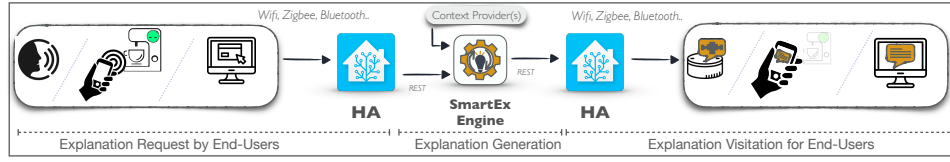---

[3]https://github.com/ExmartLab/SmartEx-Engine

Fig. 3. Explanation generation workflow

TABLE III
EXPLANATIONS OF THE SAME *explanandum* (TV IS MUTED) FOR THREE DIFFERENT USERS (BOB, ALICE, DANA)

User	Context (CEC)	View	Explanation
Bob	{State: Break, Occurrence: 1st time, Technicality: Tch, Role: **Ow**}	Fact Expl.	Hi Bob, tv_mute is active because currently a meeting in room 1 is going on and the TV is playing.
Alice	{State: Break, Occurrence: 1st time, Technicality: Tch, Role: **Cw**}	Full Expl.	Hi Alice, tv_mute is active because Bob has set up a rule: "Rule_2: mutes the TV if the TV is playing while a meeting is going on" and currently a meeting in room 1 is going on and the TV is playing, so the rule has been fired.
Dana	{State: Break, Occurrence: 1st time, Technicality: Tch, Role: **Gst**}	Simpl. Expl.	Hi Dana, Bob has set up a rule and at this moment, the rule has been fired.

plugs, and appliances working on WiFi, Zigbee, and Bluetooth. Home Assistant (HA)[4] serves as our software hub, enabling task automation via its rule-based mechanism. HA offers a RESTful API with endpoints for fetching data, including user information, automation rules, device states, and a log of past activities and states. The *Context Manager* in *SmartEx* fetches this runtime data continuously from HA, while more static data, such as the user profiles, is directly stored in the *SmartEx* database. Collecting more contextual information about the user's state (e.g., meeting) must be provided by additional services, which are not part of *SmartEx*. Additionally, the *Context Manager* stores situations that have already been explained to a user in the last three months. This information is then used by the *Context-Aware Explanation Generator* to tailor the explanation accordingly (based on the `Occurrence` context). The *Inference Function*, which determines a suitable view of the explanation based on the current context, has been implemented using the Easy-Rule[5] Java rule engine. Accordingly, the View-Context mappings policies (see Table II) are defined as rules processable by the rule engine. Lastly, the determined view is sent to the *Explanation Presentation* component, which utilizes a *Transformation Function* to generate natural language, using a template-based approach for English text generation.

The workflow in Figure 3 displays how users can seek explanations for unexpected behavior using voice commands (e.g., 'What just happened?'), NFC tag scans, or a custom HA dashboard. The first option offers a general explanation of the latest system action, while the latter enables users to specify a particular device's action. Explanations are delivered via audio and visual media, read aloud by the voice controller, and displayed on client applications such as mobile apps (Android/iOS) and web browsers

Exemplary Scenario: TV suddenly mutes. To demonstrate our prototype, we pick the motivating scenario described in Section III-A where the kitchen TV is muted during meetings in nearby rooms to prevent disruption, involving three users: Alice, Bob, and Dana.

[4]https://www.home-assistant.io/
[5]https://github.com/j-easy/easy-rules

To generate explanations, *SmartEx* collects *AEC* (see Section IV-A1) and *CEC* (see Section IV-A2). The AECs are context and user-independent. The algorithm identifies that the reason for muting the TV is that a specific rule, called *Rule_2*, was fired. *SmartEx* then uses the collected CECs to generate a context-aware explanation. Table III shows the explanations generated by *SmartEx* for the same *explanandum* (i.e., TV is muted) for the three individual users displayed on their devices.

To generate a context-aware and personalized explanation for the three users, the *Context-Aware Explanation Generator* must determine the best-suited *View* based on the CECs and the rules specified in Table II. The *Context-Aware Explanation Generator* determines the *Fact Exp.* for Bob, the *Full Exp.* for Alice, and the *Simplified Exp.* for Dana.

VI. CONCLUSION

Despite theoretical foundations and empirical findings, there's a notable lack of effective software engineering methods and technical solutions for generating user-centric explanations in smart environments. To bridge this gap, we proposed a novel mechanism and reference architecture for explanation generation in this domain. The framework generates context-aware and personalized explanations on various levels of granularity tailored to users and situations. The framework follows a service-oriented architecture that enables integration with existing pervasive smart environment systems to equip them with self-explainability functions. Our experiments on generating explanations for everyday scenarios in smart environments show the feasibility of the approach and motivate further work. In forthcoming research, we plan to expand and enrich our context model to encompass a broader range of contextual elements. Furthermore, we plan to incorporate other types of intelligent systems (recommendation systems, ML-based systems, etc.) and more complex types of explanations, such as contrastive explanations [53]. Also, we plan to make the explanation process more interactive, so users can provide feedback to the system to receive more detailed explanations.

REFERENCES

[1] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, 2018.

[2] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[3] I. Nunes and D. Jannach, "A systematic review and taxonomy of explanations in decision support and recommender systems," *User Modeling and User-Adapted Interaction*, vol. 27, no. 3, 2017.

[4] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (XAI): Toward medical XAI," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 11, pp. 4793–4813, 2020.

[5] T. Jakobi, G. Stevens, N. Castelli, C. Ogonowski, F. Schaub, N. Vindice, D. Randall, P. Tolmie, and V. Wulf, "Evolving needs in IoT control and accountability: A longitudinal study on smart home intelligibility," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 4, pp. 1–28, 2018.

[6] D. Minh, H. X. Wang, Y. F. Li, and T. N. Nguyen, "Explainable artificial intelligence: a comprehensive review," *Artificial Intelligence Review*, pp. 1–66, 2022.

[7] E. Puiutta and E. M. Veith, "Explainable reinforcement learning: A survey," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 2020, pp. 77–95.

[8] B. Y. Lim, A. K. Dey, and D. Avrahami, "Why and why not explanations improve the intelligibility of context-aware intelligent systems," in *SIGCHI Conference on Human Factors in Computing Systems (CHI)*. Association for Computing Machinery, 2009, p. 2119–2128.

[9] B. Meerbeek, T. van Druenen, M. Aarts, E. van Loenen, and E. Aarts, "Impact of blinds usage on energy consumption: automatic versus manual control," in *European Conference on Ambient Intelligence (AmI)*. Springer, 2014, pp. 158–173.

[10] M. Winikoff, "Towards trusting autonomous systems," in *International workshop on engineering multi-agent systems*. Springer, 2017, pp. 3–20.

[11] D. Gunning, "Explainable artificial intelligence (XAI)," *Defense advanced research projects agency (DARPA)*, 2017.

[12] J. Sauer and B. Rüttinger, "Automation and decision support in interactive consumer products," *Ergonomics*, vol. 50, no. 6, pp. 902–919, 2007.

[13] O. Vermasan and P. Friess, "Internet of things—from research and innovation to market development," 2014.

[14] A. Abdul, J. Vermeulen, D. Wang, B. Y. Lim, and M. Kankanhalli, "Trends and trajectories for explainable, accountable and intelligible systems," in *CHI Conf. on Human Factors in Computing Systems*, 2018.

[15] Q. V. Liao and K. R. Varshney, "Human-centered explainable AI (XAI): From algorithms to user experiences," *arXiv preprint:2110.10790*, 2021.

[16] S. T. Mueller, E. S. Veinott, R. R. Hoffman, G. Klein, L. Alam, T. Mamun, and W. J. Clancey, "Principles of explanation in human-ai systems," *arXiv preprint arXiv:2102.04972*, 2021.

[17] J. Waskan, "Mechanistic explanation at the limit," *Synthese*, vol. 183, pp. 389–408, 2011.

[18] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[19] S. Glennan, "Rethinking mechanistic explanation," *Philosophy of science*, vol. 69, no. S3, pp. S342–S353, 2002.

[20] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, 2019.

[21] P. Lipton, "Contrastive explanation," *Royal Institute of Philosophy Supplement*, vol. 27, pp. 247—266, 1990.

[22] U. Ehsan and M. O. Riedl, "Human-centered explainable AI: Towards a reflective sociotechnical approach," in *HCI International 2020*. Springer International Publishing, 2020, pp. 449–466.

[23] T. Lombrozo, "The structure and function of explanations," *Trends in cognitive sciences*, vol. 10, no. 10, pp. 464–470, 2006.

[24] D. A. Wilkenfeld and T. Lombrozo, "Inference to the best explanation (ibe) versus explaining for the best inference (ebi)," *Science & Education*, vol. 24, no. 9, pp. 1059–1077, 2015.

[25] A. Bunt, M. Lount, and C. Lauzon, "Are explanations always important?" in *ACM International Conf. on Intelligent User Interfaces*. ACM, 2012.

[26] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: Technologies, applications, and opportunities," *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 277–298, 2009.

[27] J. Huang and M. Cakmak, "Supporting mental model accuracy in trigger-action programming," in *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2015, pp. 215–225.

[28] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, "Interpretable & explorable approximations of black box models," *arXiv preprint arXiv:1707.01154*, 2017.

[29] P. Brezillon and J.-C. Pomerol, "Joint cognitive systems, cooperative systems and decision support systems: A cooperation in context," in *European Conference on Cognitive Science*, 1997, pp. 129–139.

[30] M. Li and J. Mao, "Hedonic or utilitarian? exploring the impact of communication style alignment on user's perception of virtual health advisory services," *International Journal of Information Management*, vol. 35, no. 2, pp. 229–243, 2015.

[31] R. R. Hoffman and G. Klein, "Explaining explanation, part 1: theoretical foundations," *IEEE Intelligent Systems*, vol. 32, no. 3, pp. 68–73, 2017.

[32] R. R. Hoffman, S. T. Mueller, and G. Klein, "Explaining explanation, part 2: Empirical foundations," *IEEE Intelligent Systems*, vol. 32, no. 4, 2017.

[33] G. Klein, "Explaining explanation, part 3: The causal landscape," *IEEE Intelligent Systems*, vol. 33, no. 2, pp. 83–88, 2018.

[34] B. Mittelstadt, C. Russell, and S. Wachter, "Explaining explanations in AI," in *Conference on Fairness, Accountability, and Transparency*, 2019.

[35] A. Páez, "The pragmatic turn in explainable artificial intelligence (XAI)," *Minds and Machines*, vol. 29, no. 3, 2019.

[36] S. Gregor and I. Benbasat, "Explanations from intelligent systems: Theoretical foundations and implications for practice," *MIS quarterly*, pp. 497–530, 1999.

[37] V. Zhao, L. Zhang, B. Wang, M. L. Littman, S. Lu, and B. Ur, "Understanding trigger-action programs through novel visualizations of program differences," in *CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–17.

[38] E. Houzé, A. Diaconescu, J.-L. Dessalles, and D. Menga, "A generic and modular reference architecture for self-explainable smart homes," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, 2022.

[39] B. Kordts, B. Gerlach, and A. Schrader, "Towards self-explaining ambient applications," in *14th PErvasive Technologies Related to Assistive Environments Conference (PETRA)*. ACM, 2021, pp. 383–390.

[40] D. Burmeister, F. Burmann, and A. Schrader, "The smart object description language: Modeling interaction capabilities for self-reflection," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2017.

[41] M. Blumreiter, J. Greenyer, F. Chiyah Garcia, V. Klös, M. Schwammberger, C. Sommer, A. Vogelsang, and A. Wortmann, "Towards self-explainable cyber-physical systems," in *ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. IEEE, 2019.

[42] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[43] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[44] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *IEEE 5th International Conference on data science and advanced analytics (DSAA)*. IEEE, 2018, pp. 80–89.

[45] B. Neuhofer, D. Buhalis, and A. Ladkin, "Smart technologies for personalized experiences: a case study in the hospitality domain," *Electronic Markets*, vol. 25, no. 3, pp. 243–254, 2015.

[46] S. Alletto, R. Cucchiara, G. Del Fiore, L. Mainetti, V. Mighali, L. Patrono, and G. Serra, "An indoor location-aware system for an IoT-based smart museum," *IEEE Internet of Things Journal*, vol. 3, no. 2, 2016.

[47] L. Baresi, M. Sadeghi, and M. Valla, "TDeX: A description model for heterogeneous smart devices and GUI generation," in *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018.

[48] M. Ehatisham-ul Haq and M. A. Azam, "Opportunistic sensing for inferring in-the-wild human contexts based on activity pattern recognition using smart computing," *Future Generation Com. Systems*, 2020.

[49] M. Sadeghi, V. Klös, and A. Vogelsang, "Cases for explainable software systems: characteristics and examples," in *IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 2021.

[50] J. Y. Halpern and J. Pearl, "Causes and explanations: A structural-model approach. part ii: Explanations," *The British journal for the philosophy of science*, 2005.

[51] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 414–454, 2013.

[52] B. Y. Lim and A. K. Dey, "Toolkit to support intelligibility in context-aware applications," in *12th ACM International Conference on Ubiquitous Computing (UbiComp)*. ACM, 2010, pp. 13–22.

[53] L. Herbold, M. Sadeghi, and A. Vogelsang, "Generating context-aware contrastive explanations in rule-based systems," in *IEEE/ACM International Workshop on Explainability Engineering (ExEn)*, 2024.